НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: __Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних__
_____
_____

Виконав: студент IV курсу, групи __КВ-51__
(шифр групи)

_____Тимошенко Ігор Олегович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ___доц.каф.СПСКС, к.т.н. Петрашенко А.В _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
_____Тимошенка Ігоря Олеговича_____
(прізвище, ім’я, по батькові)

1. Тема проекту _Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних _____
_____,
керівник проекту ____доц.каф. СПСКС, к.т.н. Петрашенко А.В._____ ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

- Аналіз існуючих рішень та обґрунтування теми дипломного проекту
- Структура розроблених програмних засобів
- Алгоритм аналізу потокових даних
- Використання розроблених програмних засобів та аналіз результатів роботи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Архітектура програмного засобу. Схема структурна
- Робочий потік даних. Схема алгоритму
- Аналіз даних в контексті Apache Spark. Схема алгоритму
- Структура сервісу обробки даних. Схема структурна

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Кдятченко Я.М. доцент		

7. Дата видачі завдання «___» _____2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	15.04.2019	
2.	Розроблення та узгодження технічного завдання	30.04.2019	
3.	Аналіз існуючих рішень	05.05.2019	
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019	
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019	
6.	Підготовка графічної частини дипломного проекту	20.05.2019	
7.	Оформлення документації дипломного проекту	25.05.2019	
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Тимошенко І.О.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Петрашенко А.В.
(підпис) (ініціали, прізвище)

Анотація

Метою дипломного проекту є розробка веб-орієнтованого програмного засобу аналізу потокових сенсорних даних, а саме – забезпечення виявлення об’єктів на кадрах користувацької відеокамери в режимі реального часу.

Для виконання проекту були проаналізовані існуючі рішення виявлення об’єктів на відео та зображеннях засобами машинного навчання і нейронних мереж, а також переглянуті існуючі методи ефективної роботи з потоковими даними. На їх основі був розроблений новий продукт, що дозволяє:

- через веб-сторінку в браузері слідкувати за наявністю об’єктів в рамках видимості відеокамери та отримувати візуальне виділення їх місцезнаходження;
- користування розробленою системою окремо від веб-інтерфейсу, засобами спілкування через комп’ютерні мережі;
- швидкий та точний аналіз потокових даних.

Основною мовою розробки стала мова програмування Python, яка дає доступ до широкого набору інструментів роботи з алгоритмами машинного навчання, обробки зображень та засобами розподілених обчислень. Для серверної частини веб-інтерфейсу використаний Node.js.

Ключові слова: згорткові нейронні мережі, відео виявлення об’єктів, потоковий аналіз даних, розподілені обчислення, Apache Spark, Python.

Annotation

The purpose of the diploma project is the development of a web-based software for analysis of stream sensory data, namely - ensuring the detection of objects on the frames of a user's video camera in real time.

For the implementation of the project, were analyzed existing solutions for detecting objects on video and images by means of machine learning and neural network, as well as revised existing methods of efficient work with stream data. On their basis, was developed a new product, which allows:

- through the web page in the browser to monitor the presence of objects in the visibility of the video camera and receive a visual allocation of their location;
- use of the developed system separately from the web interface, means of communication through computer networks;
- fast and accurate analysis of streaming data.

The main language of development was the Python programming language, which provides access to a wide range of tools for working with algorithms for machine learning, image processing and distributed computing. Node.js is used for the server part of the web interface.

Keywords: Convolutional Neural Networks, Video Object Detection, Data Flow Analysis, Distributed Computing, Apache Spark, Python.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045440.002 ТЗ	Веб-орієнтовані програмні	4		
			засоби аналізу потокових			
			сенсорних даних			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Веб-орієнтовані програмні	2		
			засоби аналізу потокових			
			сенсорних даних			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045430.004 ПЗ	Веб-орієнтовані програмні	50		
			засоби аналізу потокових			
			сенсорних даних			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Тимошенко			Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко					
Консульт.						НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-51	
Н. контроль		Клятченко					
Зав. каф.		Тарасенко					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Архітектура програмного			
			засобу. Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Робочий потік даних.			
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Аналіз даних в контексті			
			Apache Spark. Схема			
			алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Структура сервісу			
			обробки даних. Схема			
			структурна			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних».

Галузь застосування: системи відеоспостереження, охоронні системи, системи збору статистичних даних, системи автопілотів.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою дипломного проекту є розробка веб-орієнтованого програмного засобу аналізу потокових сенсорних даних, а саме – забезпечення виявлення обє'ктів на кадрах користувацької відеокамери в режимі реального часу.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- достатня точність виявлення;
- низька або зовсім відсутня кількість хибних виявлень;
- висока швидкість обробки даних відеопотоку при збереженні необхідного рівня точності;
- зрозумілий та візуально простий веб-інтерфейс

5.2. Вимоги до апаратного забезпечення

- процесор: Intel Core i7 Skylake або вище;
- мінімальний об`єм оперативної пам’яті: 8 Гб;
- графічний процесор: Nvidia GTX 980 або краще;

5.3. Вимоги до програмного та апаратного забезпечення користувача

- будь-який браузер, крім Internet Explorer та Safari;
- будь-яка операційна система з можливістю користування браузером;
- наявність підключеної камери до користувацької системи;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045430.004 ПЗ	Веб-орієнтовані програмні	50		
			засоби аналізу потокових			
			сенсорних даних			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Архітектура програмного			
			засобу. Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Робочий потік даних.			
			Схема алгоритму			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Тимошенко			Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Петрашенко					
Консульт.						НТУУ «КПІ ім. І. Сікорського», ФПМ, КВ-51	
Н. контроль		Клятченко					
Зав. каф.		Тарасенко					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.007 Д3	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Аналіз даних в контексті			
			Apache Spark. Схема			
			алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Веб-орієнтовані програмні	1		
			засоби аналізу потокових			
			сенсорних даних			
			Структура сервісу			
			обробки даних. Схема			
			структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП	Арк. 2

ЗМІСТ

ДОДАТКИ

Додаток 1. Копії графічних матеріалів

- ІАЛЦ.045440.005 Д1. Архітектура програмного засобу. Схема структурна
- ІАЛЦ.045440.006 Д2. Робочий потік даних. Схема алгоритму
- ІАЛЦ.045440.007 Д3. Аналіз даних в контексті Apache Spark. Схема алгоритму
- ІАЛЦ.045440.008 Д4. Структура сервісу обробки даних. Схема структурна

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

AP (Average Precision) – метрика, що визначає характеристику точності та повноти класифікатора.

API (Application programming interface) – набір визначених методів для взаємодії різних компонентів.

CNN (Convolutional neural network) – клас глибинних нейронних мереж прямого поширення, який застосовувався до аналізу візуальних зображень.

COCO (Common Objects in Context) – великий набір даних для задач комп’ютерного зору.

CPU (Central process unit) – центральний процесор.

DNN (Deep Neural Network) – штучна нейронна мережа з декількома прихованими шарами вузлів між вхідним та вихідним шарами.

DPM (Deformable parts model) – метод аналізу даних для класифікації та регресійного аналізу.

FC (Full connected) – скорочення що описує метод з’єднання шарів штучної нейронної мережі.

FPS (Frame per second) – кількість кадрів на секунду.

HOG (Histogram of oriented gradients) – дескриптор ознак, який використовується в комп'ютерному зорі і обробці зображень з метою розпізнання об'єктів.

JSON (JavaScript Object Notation) – формат представлення даних, що за своєю структурою нагадує структуру об’єктів мови JavaScript.

R-CNN (Regions with CNNs) – підхід до виявлення об’єктів з застосуванням виділення рекомендаційних регіонів перед обробкою в згорткових нейронних мережах.

RDD (Resilient Distributed Data) – структура даних, що є фундаментальною для Apache Spark.

RoI (Region of Interest) – додатковий шар згорткової нейронної мережі, що використовується в алгоритмі Fast R-CNN.

RPN (Region Proposal Network) – окрема мережа для визначення пропозицій регіону введена в підході Faster R-CNN.

SIFT (Scale-invariant feature transform) – алгоритм із області комп'ютерного зору, який виявляє і описує локальні ознаки зображення.

SVM (Support-vector machines) – метод аналізу даних для класифікації та регресійного аналізу.

TCP (Transport Control Protocol) – протокол транспортного рівня.

UDP (User Datagram Protocol) – протокол транспортного рівня.

YOLO (You Only Look Once) – принцип виявлення об'єктів за допомогою регресії з фіксованою сіткою.

ВСТУП

Аналіз потокових даних в сучасній сфері інформаційних технологій має дуже важливе значення для функціонування більшості програмних засобів та сервісів світових лідерів індустрії. Обробка даних про діяльність користувачів, аналіз стану віддалених систем або ж обробка постійного потоку даних користувачів має відбуватися якісно, а головне – швидко.

Говорячи про потоковий аналіз даних, головними джерелами їх надходження в першу чергу слід вважати різні види сенсорів, що постійно надають нову інформацію в режимі «без зупинок». Такими сенсорами можуть бути як відео- та аудіо- записні пристрої, так і сенсори фіксації мережного трафіку, температур тощо.

Однією з перспективних областей аналізу даних за допомогою відео сенсорів стала область виявлення об'єктів на зображеннях та відео. Дана область не так давно отримала величезний прогрес, в основному завдяки появі та розвитку глибинного навчання. Коротко кажучи, виявлення об'єктів – це завдання ідентифікації об'єктів на зображенні та їх локалізація. Це дуже важлива проблема в комп'ютерному зорі завдяки своїм численним застосуванням від простого відеоспостереження та сучасних систем автопілоту до систем безпеки та вистежування.

Відображаючи вимоги індустрії в аналізі великої кількості даних в умовах реального часу на дану область, виникає потреба у вдосконаленні вже існуючих рішень. Системи яким необхідно постійно оцінювати стан навколишнього середовища або слідкувати за рухом об’єктів, не можуть собі дозволити обробку одного кадру відео потоку більше ніж сотню мілісекунд. Це означає, що необхідно розробляти нові системи та алгоритми, які зможуть збільшити швидкість аналізу зображень при цьому не наносячи великих втрат в характеристиці точності ідентифікації об’єктів.

Для вирішення цієї проблеми в дипломному проекті було взято за мету до існуючих алгоритмів і методів комп’ютерного зору застосувати відомі підходи аналізу великих даних (Big Data). Зокрема використати головний принцип цих підходів – розподілення обчислень між вузлами серверного кластеру. Подібне рішення може забезпечити покращення параметрів швидкодії аналізу потоку даних із збереженням встановлених критерій області виявлення об’єктів та надати можливість збільшення навантаження на систему за рахунок відповідного розширення кластеру.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Загальний опис проблеми виявлення об’єктів

Задача виявлення об'єктів полягає в тому, щоб визначити, де на заданому зображенні розташовані об'єкти (локалізація) і до якої категорії кожен об'єкт належить (класифікація). Таким чином, порядок дій традиційних моделей виявлення об'єктів можна розділити в основному на три етапи: вибір інформативної області, виділення ознак і класифікація [1].

Вибір інформативної області**.** Якщо різні об'єкти можуть з'являтися в будь-яких місцях зображення і мають різні розміри та їх співвідношення, то природним вибором є сканування всього зображення за допомогою різномасштабних ковзаючих вікон. Хоча ця вичерпна стратегія виявляє всі можливі позиції об'єктів, її недоліки також очевидні. Через велику кількість можливих варіантів вікон, цей підхід є затратним в обчислюваннях і виробляє надлишкову кількість вікон. Але, якщо застосовувати тільки фіксоване число шаблонів ковзаючих вікон, можуть бути вироблені незадовільні області.

Виділення ознак. Щоб розпізнати різні об'єкти, необхідно виділити візуальні особливості, які можуть забезпечити семантично повне представлення об’єкту. Масштабонезалежне перетворення ознак (SIFT), гістограма напрямлених градієнтів (HOG) [23] і ознаки Хаара [25] є найбільш репрезентативними варіантами. Це пов'язано з тим, що ці ознаки можуть виробляти уявлення, пов'язані з комплексними клітинами мозку людини. Проте, через різноманітність видів, умов освітлення та фонів, важко вручну розробити надійний дескриптор ознак для досконалого опису всіх видів об'єктів.

Класифікація. Далі, необхідний класифікатор для розрізнення цільового об'єкта від усіх інших категорій і виконання його представлення більш ієрархічним, семантичним та інформативним для візуального розпізнавання.

Зазвичай, хорошим вибором є метод опорно-векторних машин (SVM) [17], AdaBoost і модель на основі деформованих деталей (DPM) [18]. Серед цих класифікаторів, DPM є гнучкою моделлю, яка за рахунок деформацій об’єднує частин об'єкта для усунення суворих деформацій. У DPM, за допомогою графічної моделі ретельно розроблені низькорівневі ознаки і декомпозиції кінематично натхненних частин об'єднуються. І дискримінантне вивчення графічних моделей дозволяє будувати високоточні моделі на основі деталей для різноманітних класів об'єктів.

На основі цих дискримінантних локальних ознак і дрібних архітектур, були отримані хороші результати досліджень на конкурсі виявлення об’єктів PASCAL VOC [20], а вбудовані системи реального часу були отримані з низьким навантаженням на обладнання. Але, разом з тим, в 2010-2012 рр. невеликі вигоди були отримані лише за рахунок побудови цілісних систем і використання незначних варіантів покращень успішних методів.

Цей факт обумовлений наступними причинами:

1) Генерація обмежуючих рамок за стратегією ковзаючого вікна є надмірною, неефективною та неточною.

2) Семантичний розрив не може бути усунутий комбінацією в ручну розроблених дескрипторів низького рівня та диференційно-навчених невеликих моделей.

Завдяки неймовірному розвитку глибинних нейронних мереж (DNN) та з введенням Regions with CNNs (R-CNN) [2] було отримане більш значне посилення. DNN, або найбільш представницькі згорткові нейронні мережі (CNN), діють зовсім інакше, ніж традиційні підходи. Вони мають більш глибокі архітектури з можливостями виокремлення більш складних ознак, ніж неглибокі. Крім того, виразні і надійні алгоритми тренування дозволяють отримати інформативні уявлення про об'єкти без необхідності розробки ознак вручну.

З моменту пропозиції R-CNN було запропоновано багато вдосконалених моделей, у тому числі Fast R-CNN, який одночасно оптимізує класифікаційні та обмежувальні регресійні завдання, Faster R-CNN, що використовує додаткову підмережу для створення пропозицій регіонів і YOLO який здійснює виявлення об'єктів за допомогою регресії з фіксованою сіткою. Всі вони приносять різні ступені вдосконалення виявлення ознак над первинним R-CNN і роблять точне виявлення об'єктів в реальному часі більш досяжним.

1.2. Аналіз сучасних підходів

Дана задача спрямована на виявлення і класифікацію існуючих об'єктів на певному зображенні і маркування їх за допомогою прямокутних обмежувальних рамок, щоб показати конфіденційність існування. Як було наведено вище, сучасні алгоритми в основному базуються на принципах використання глубинних нейронних мереж. Але в залежності від загальних методів виявлення об'єктів, підходи можуть бути розділені на два типи:

1) ті, що слідкують за традиційним конвеєром виявлення об'єктів, спочатку генеруючи пропозиції регіону, а потім класифікуючи кожну пропозицію в різні категорії об'єктів;
2) ті, що розглядають виявлення об'єктів як проблему регресії/класифікації, що приймає уніфіковані рамки для досягнення кінцевих результатів (категорій та місцеположень) безпосередньо.

Методи, що базуються на пропозиціях регіонів, включають переважно R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, Mask R-CNN, деякі з яких корелюють один з одним. Методи регресії/класифікації в основному включають MultiBox, AttentionNet, G-CNN, YOLO, SSD [24], DSSD і DSOD. Кореляції між цими двома загальними методиками пов’язані принципами, впровадженими в Faster R-CNN.

Методи, що базуються на пропозиціях регіонів

Структура, що базується на пропозиціях регіонів, двоетапний процес, відповідає механізму уваги людського мозку до деякого ступеня, що дає грубу перевірку всього сценарію, а потім зосереджує увагу на регіонах, що цікавлять.

R-CNN. Важливе значення має поліпшення якості обмежувальних коробок кандидатів і глибокої архітектури для вилучення ознак високого рівня. Для вирішення цих проблем, був запропонований R-CNN Россом Гіршиком в 2014 році і отримав середню точність (mAP) 53,3% з більш ніж 30% поліпшенням порівняно з попереднім найкращим результатом [2]. На рис. 1.1 показана блок-схема R-CNN, яка може бути розділена на три етапи наступним чином.

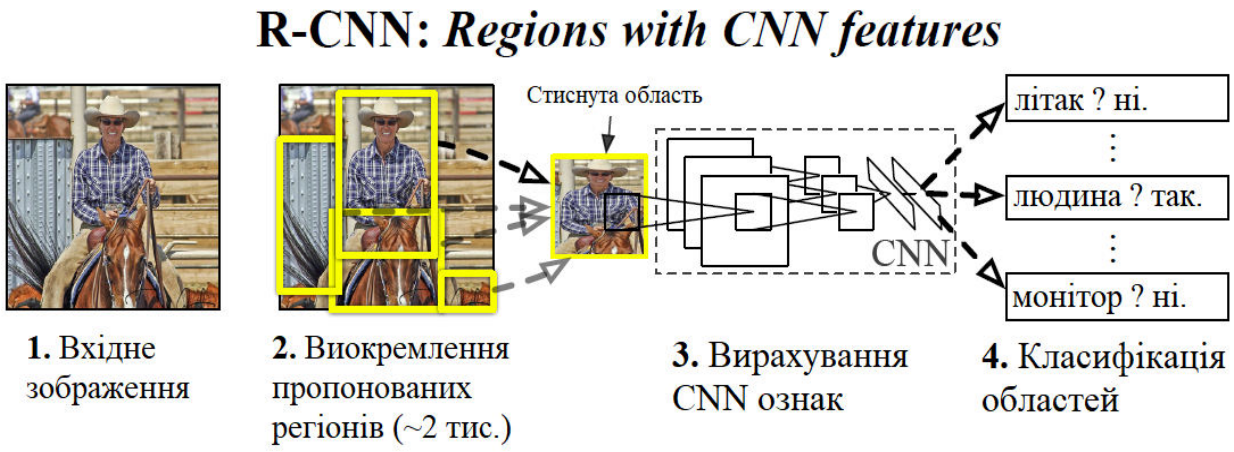


Рисунок 1.1 – Узагальнена послідовність дій алгоритму R-CNN

Генерація пропозиції регіону. R-CNN приймає вибірковий пошук [21] для генерування близько 2000 пропозицій для кожного зображення. Метод селективного пошуку покладається на просте групування знизу-вгору і

відмінності, щоб швидко забезпечити більш точні рамки-кандидати довільних розмірів і зменшити простір пошуку для виявлення об’єктів.

Глибинне виділення ознак на основі CNN. На цій стадії кожна пропозиція регіону згортається або обрізається до фіксованого розміру, а внутрішня частина модулю CNN використовується для вилучення 4096-мірної ознаки як остаточного представлення. Через велику здатність до навчання, домінуючу виразну потужність і ієрархічну структуру CNN, можна отримати високо рівневі, семантичні та надійні представлення ознак для кожної пропозиції регіону.

Класифікація та локалізація. З попередньо навченими на специфічні категорії лінійними SVM для декількох класів, запропоновані регіони оцінюються на множині позитивних областей і фонових (негативних) областей. Оцінені області потім регулюються за допомогою регресії обмежувальної рамки і фільтруються за допомогою методу не максимального придушення для отримання кінцевих обмежувальних рамок для збережених розташувань об'єктів.

Незважаючи на свої вдосконалення в порівнянні з традиційними методами і значущістю приведення CNN до практичного виявлення об'єктів, є ще деякі недоліки:

- внаслідок існування повнозв’язних шарів, CNN вимагає вхідного зображення фіксованого розміру (наприклад, 227×227), що безпосередньо призводить до повторного переобчислення всього CNN для кожного оцінюваного регіону, займаючи багато часу в період тестування;
- навчання R-CNN є багатоступеневим конвеєром. По-перше, згорткова мережа (CNN) на об'єктних пропозиціях точно налаштовується. Потім класифікатор, отриманий за допомогою тонкого налаштування, замінюється на SVM, щоб забезпечити роботу з ознаками CNN. І нарешті, налаштування регресії обмежуючих рамок;

- хоча вибірковий пошук може генерувати пропозиції регіону з відносно високими задовольняючими результатами, отримані пропозиції регіону залишаються надлишковими, і ця процедура вимагає багато часу (близько 2 секунд для видобування 2000 пропозиції регіону);
- методика не може використовуватися в реальному часі, так як це займає приблизно 42 секунд на одне зображення.

Fast R-CNN: З метою усунення деяких з вищезгаданих недоліків методу R-CNN, автор R-CNN запропоновував введення багатозадачної втрати на класифікацію і регресію обмежуючих рамок, і запропонував нову архітектуру CNN під назвою Fast R-CNN [3]. Архітектура Fast R-CNN представлена на рис. 1.2.

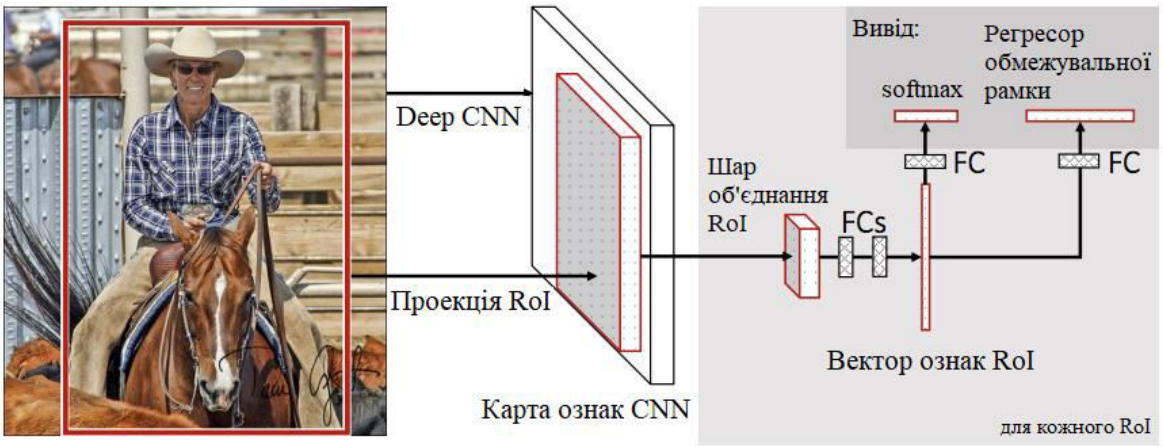


Рисунок 1.2 – Схема алгоритму Fast R-CNN

Ціле зображення обробляється за допомогою згорткових шарів для створення карти об'єктів. Потім з кожного запропонованого регіону витягується вектор ознак фіксованої довжини за допомогою шару об’єднання Region of Interest (RoI). Кожен вектор ознак потім подається в послідовність повнозв’язних шарів (FC) перед тим, як остаточно розгалужиться на два вихідних шари. Один вихідний шар відповідає за виробництво softmax

ймовірностей для всіх C + 1 категорій (C класів об’єктів плюс один «фоновий» клас), а інший вихідний шар кодує уточнені позиції обмежувальної рамки з чотирма раціональними числами. Всі параметри в цих процедурах (за винятком генерації пропозицій регіонів) оптимізуються через багатозадачні втрати в наскрізному проході.

У Fast R-CNN, незалежно від генерації пропозиції регіону, навчання всіх мережевих шарів може бути оброблено в один етап з багатозадачною втратою. Це економить додаткові витрати на зберігання в пам’яті, а також підвищує точність та ефективність в застосуванні з більш кращими схемами навчання.

Faster R-CNN. Обидва вищезгадані алгоритми R-CNN та Fast R-CNN використовують вибірковий пошук, щоб з'ясувати пропозиції регіону. Вибірковий пошук є повільним і трудомістким процесом, що впливає на продуктивність мережі. Тому був придуманий алгоритм виявлення об'єктів, який виключає алгоритм вибіркового пошуку і дозволяє окремій мережі пропозиції регіону (RPN) формувати ці пропозиції [4].

Подібно до Fast R-CNN, зображення подається в якості вхідного для згорткової мережі, яка забезпечує визначення карти ознак. Замість використання алгоритму вибіркового пошуку на карті ознак для ідентифікації пропозицій регіону, для прогнозування пропозицій регіону використовується окрема мережа RPN. Запропоновані пропозиції регіону змінюють, використовуючи шар об'єднання RoI, який потім використовують для класифікації зображення в межах запропонованої області і прогнозування значень зміщення для обмежувальних рамок. Даний процес зображений на рис.1.3.

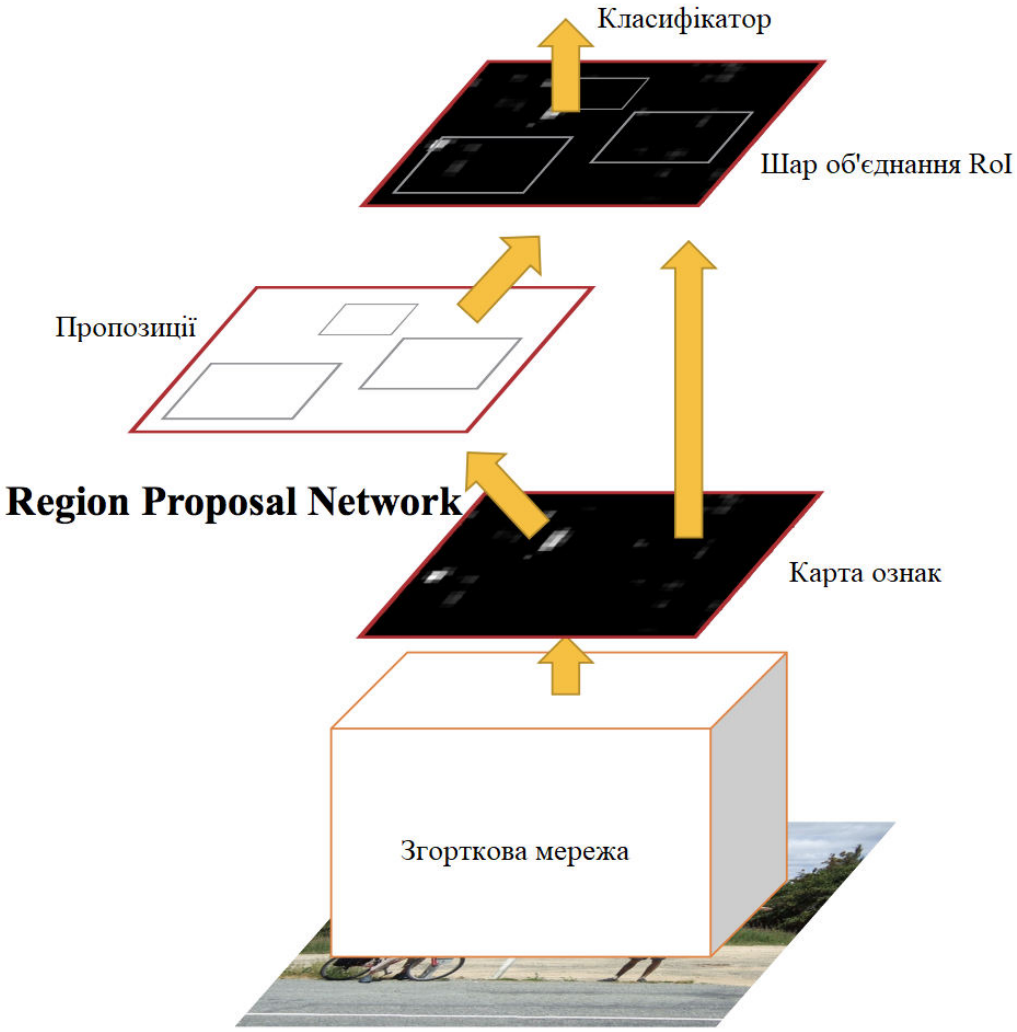


Рисунок 1.3 – Принцип роботи Faster R-CNN

Методи регресії/класифікації

Принципи, що базуються на регіональних пропозиціях, складаються з декількох взаємозалежних етапів, включаючи генерування пропозицій регіонів, вилучення ознак з CNN, класифікацію та регресію обмежувальної рамки, які зазвичай навчаються окремо. Навіть у нещодавньому наскрізному модулі Faster R-CNN необхідна альтернативна підготовка для отримання загальних параметрів згортки між RPN і мережею виявлення. Як результат, час, витрачений на обробку різних компонентів, стає вузьким місцем у додатках з обробкою в реальному режимі часу.

Одноетапні підходи, основані на глобальній регресії/класифікації можуть зменшити витрати часу, відображаючи прямо з пікселів зображення на обмежувальні координати і ймовірності класу. Далі наведений один з найпопулярніших варіантів цього підходу.

YOLO. Джозеф Редмон запропонував новий принцип під назвою «You Look Only Once» (YOLO) [5], який використовує всю верхню карту ознак для прогнозування одночасно впевненості для категорій та обмежувальних рамок. Основна ідея YOLO представлена на рис. 1.3.

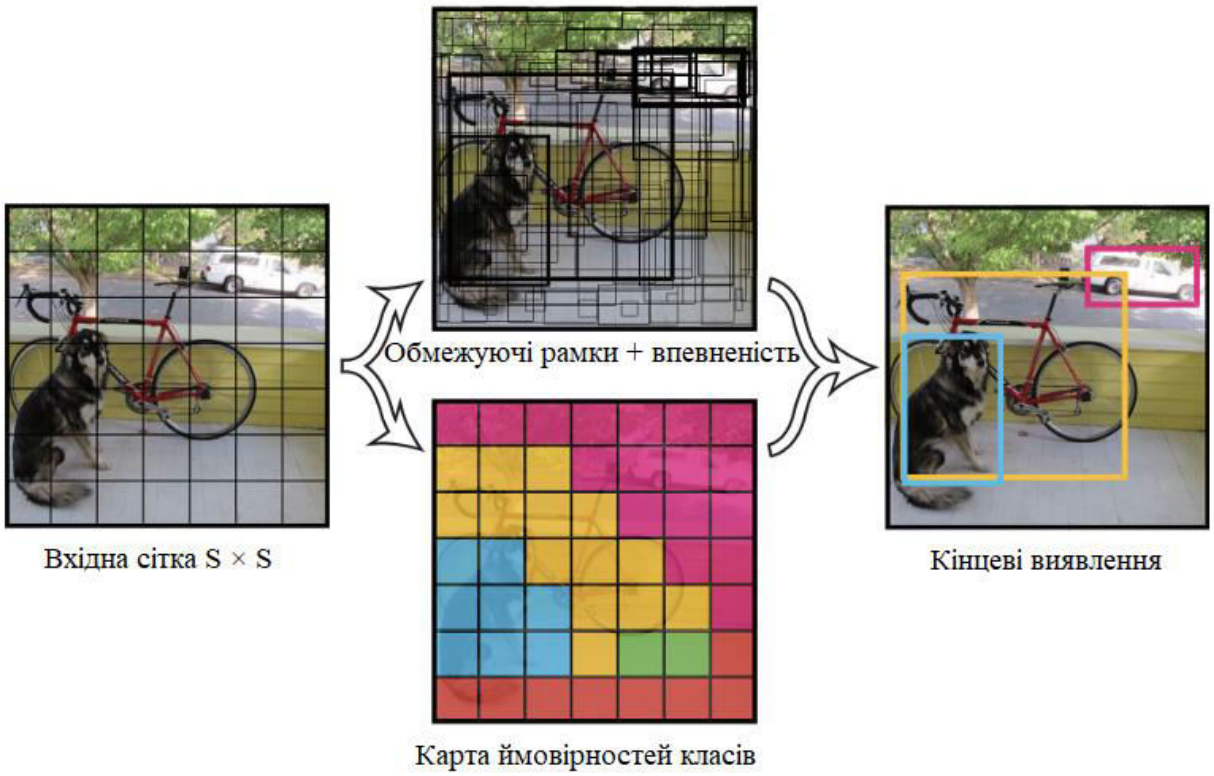


Рисунок 1.4 – Принцип роботи YOLO

YOLO ділить вхідне зображення на сітку S × S, і кожна комірка сітки відповідає за прогнозування об'єкта центрованого в цій комірці сітки. Кожна комірка сітки прогнозує обмежувальні рамки B та їх відповідні оцінки впевненості. Формально, оцінки впевненості визначаються як множення ймовірності умовного класу на впевненість прогнозу окремих комірок, що

вказує на ймовірність існування об'єктів і показує впевненість його прогнозу. У той же час, незалежно від кількості рамок, ймовірність умовного класу C також повинна бути передбачена в кожній комірці сітки. Слід зауважити, що обчислюється лише внесок з комірки сітки, що містить об'єкт.

YOLO складається з 24 згорткових і 2 повнозв’язних шарів, з яких деякі конвеєрні шари будують об’єднання початкових модулів з 1 × 1 редукційними шарами, за якими слідують 3 × 3 згорткових шари. Мережа може обробляти зображення в режимі реального часу при 45 FPS і спрощена версія Fast YOLO може досягати 155 FPS з кращими результатами, ніж інші детектори реального часу [5]. Крім того, YOLO виробляє менше помилкових спрацьовувань на задньому фоні, що робить можливим співпрацю з Fast R-CNN. Також впровадженні покращені версії, YOLOv2 [6] та найновіша YOLOv3 [7], яка приймає кілька вражаючих стратегій, що покращують показники ще більше.

Серед недоліків, у YOLO виникають труднощі при роботі з малими об'єктами в групах, що обумовлено сильними просторовими обмеженнями, що накладаються на обмежувальні рамки прогнозів.

1.3. Обґрунтування вибору теми та постановка задачі

Проаналізовані сучасні методи на основі згорткових нейронних мереж чудово справляються з задачею виявлення об’єктів. Але тільки невелика кількість серед них може бути використана для роботи в реальному часі. Але саме цього наразі потребує велика кількість комерційних та відкритих проектів широкого спектру галузей.

Підходи Faster R-CNN та YOLO виявилися найкращими кандидатами на використання в ніші аналізу потокових зображень, або ж відео потоку. Порівнюючи в цьому розділі роботу перелічених методик виявлено, що швидшим і інноваційнішим є підхід YOLO, через його економність та принцип «Ти дивишся тільки один раз». Хоча перша версія даного алгоритму

поступалася в точності перед Faster R-CNN, з часом, після виходу другої та третьої версії ці показники тримаються майже на одному рівні. З цих причин YOLO та його модифікації використовуються в багатьох сучасних розробках.

Варто зазначити, що аналіз зображень методами глибинних нейронних мереж, включаючи YOLO, є процесом, що потребує доволі великих затрат обчислювальних можливостей апаратних ресурсів. Хоча сучасні алгоритми і забезпечують певні оптимізації, але це не зменшує вимог до характеристик машин на яких виконуються обчислення. Також одними алгоритмічними методами не можливо обробляти вхідні потокові дані одразу з декількох джерел їх надходження.

Для усунення завад швидкодії обробки даних на апаратних машинах не достатньої потужності, або ж для прискорення роботи на передових апаратних засобах, необхідно розробити систему або системи, що дозволять певним чином збільшити її обчислювальні можливості. Такими системами можуть бути системи з розподіленими обчисленнями, що є популярною практикою в обробці великих даних. Беручи до уваги, що неперервний потік відео даних для аналізу також можна розглядати як навантаження у вигляді великих даних, то ідея застосування подібних обчислень стала цікавою в плані використання з задачами комп’ютерного зору.

Тому за мету дипломного проекту встановлено розробити систему, що відкриє всі переваги розподіленого аналізу для галузі виявлення об’єктів.

Для продуктивної і якісної розробки програмного засобу за вибраною темою, необхідно встановити конкретні цілі та вимоги до продукту, що має бути розроблений. Такими основними вимогами, що були сформовані на основі досліджень існуючих рішень та цілей розробки є:

- використання в якості методу виявлення об’єктів на зображеннях найшвидшого підходу – YOLO;
- забезпечення засобів передачі даних для їх обробки в режимі реального часу;

- використання розподілених обчислень для забезпечення високих показників ефективності навіть на системах середнього класу;
- надання можливості використання сервісу як повноцінного програмного засобу.

2. СТРУКТУРА РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

2.1. Загальна структура програмного засобу

За встановленими вимогами, кінцевим продуктом дипломного проекту повинен бути повноцінний програмний засіб, що забезпечує користування розробленою системою виявлення об’єктів у реальному часі.

Для забезпечення його повноцінності була використана відома структура «клієнт-сервер», компоненти якої конкретно в даному проекті мають такі значення:

1) клієнт – інтерфейс для використання розробленого засобу;
2) сервер – сервіс, що безпосередньо реалізує потоковий аналіз даних та забезпечує розподіленні обчислення.

Починаючи з інтерфейсу користувача не можна не згадати про те, що візуальне й інтерактивне представлення програмного засобу грає дуже важливу роль в його успішності. Розробка інтерфейсу повинна базуватися в першу чергу на наступних принципах:

- зручності;
- зрозумілості;
- функціональності;
- інформативності;
- простоті.

Керуючись переліченими критеріями, в проекті було вирішено використовувати простий графічний інтерфейс у вигляді веб-сторінки. Окрім варіанту веб-інтерфейсу також розглядалися інші види реалізації клієнту, зокрема:

- створення мобільного додатку;
- створення окремого програмного засобу для використання на персональних комп’ютерах.

Кожен із перелічених варіантів, включаючи веб-інтерфейс, відповідно має свої переваги та недоліки: інтерфейс у вигляді мобільного додатку є комфортним, має можливості використання в умовах пересування та повсякденного життя; створення клієнту для персональних комп'ютерів дозволяє надати швидкий та стабільний доступ до функціоналу програмного засобу в умовах стаціонарної роботи, але є обмежений своєю стаціонарністю.

Веб-орієнтований інтерфейс в свою чергу поєдную переваги перших двох реалізацій, дозволяє використання функціоналу системи не залежно від платформи та не потребує попередніх завантажень і налаштувань. Все що необхідно – це перейти за посиланням у веб-браузері, який вже скоріше за все встановлений в системі користувача.

Вибір конкретної реалізації залежить від вибору цільової платформи та умов застосування розроблених програмних засобів. Для цього дипломного проекту доцільним стало використання саме веб-орієнтованого інтерфейсу, так як він простий в плані використання та розробки.

Серверною складовою структури проекту є його головна частина, можна сказати, ядро програмного засобу, – це система виявлення об'єктів у режимі реального часу.

Дана складова є повнофункціональною системою з закінченим циклом обробки даних, що не залежить від конкретного джерела надходження відео потоку даних. Такий підхід дозволяє використання системи як сервісу не прив'язаного до інтерфейсу, що може бути використаний в комерційних цілях у вигляді платного сервісу для надання послуг відео виявлення, спостереження або інших специфічних рішень.

Складається сервер з окремих логічних модулів та сторонніх фреймворків, що забезпечують разом конвеєр обробки даних на розподілених кластерах.

Так як фактично за визначеною архітектурою передача даних між клієнтською та серверною частинами відбувається через комп'ютерні мережі,

то необхідно використовувати засоби і мережні протоколи, що задовольняють виконанню встановлених до програмного засобу вимог. Серед яких є пункт про забезпечення постійної передачі даних в режимі реального часу.

Якщо мова йде про потокову передачу даних, можливо і необхідно використовувати один з наступних мережних протоколів:

1) User Datagram Protocol (UDP) – протокол транспортного рівня для швидкої передачі даних мережею Інтернет у вигляді датаграм. Не надійний та не забезпечує збереження всіх даних та послідовності їх надходження;
2) Transport Control Protocol (TCP) – протокол транспортного рівня, що усуває недоліки UDP, але є повільнішим через додаткові процедури з'єднання, підтвердження отримання даних та перевірки на помилки.

Виходячи з потреби в швидкісній передачі кадрів відео потоку, логічним було б обрати використання UDP протоколу. Хоча він і не гарантує цілісності переданих даних, але невелика їх втрата може бути допустимою, і мізерне пошкодження зображення скоріше за все не вплине на кінцевий результат його аналізу. Цей вибір є очевидним, якщо б не було деяких інших факторів.

Такими факторами є сам процес розробки та інструментарії що використовується. Зокрема в проекті застосований фреймворк Apache Spark та його бібліотека Spark Streaming, що дозволяє отримувати дані для їх обробки у середовищі фреймворку саме через TCP з'єднання. З цих причин вибір був зроблений в сторону TCP протоколу. Детальніше про Apache Spark та його застосування далі.

2.2. Використані технології та інструменти

Як було зазначено раніше, для забезпечення взаємодії з розробленим сервісом був використаний веб-орієнтовний інтерфейс що представляє собою веб-сторінку в браузері. Ця частина проекту була названа умовно клієнтом і в

реальності поділяється на дві частини – інтерфейс та сервер що його обслуговує.

В якості сервера був використаний Node.js. Це відкритий проект, що представляє собою міжплатформне середовище виконання коду мови JavaScript поза межами браузера. На сьогоднішній день Node.js має дуже велику популярність завдяки асинхронному підходу JavaScript, швидкості та великій кількості бібліотек та модулів, що доступні для їх легкого використання.

На основі Node.js був використаний фреймворк express, один з розроблених для даної платформи модулів, що значно спрощує використання вбудованих методів роботи з HTTP протоколом.

Даний сервер в розробленому програмному засобі виконує тільки функції запуску та підтримки працездатності веб-інтерфейсу.

Другою складовою клієнтської частини є сам інтерфейс, що представлений у вигляді сторінки-додатку. Для забезпечення односторінковості був використаний принцип реактивного програмування. На відміну від класичного підходу де маніпуляції користувача з елементами сторінки призводять до завантаження або перезавантаження сторінки, в реактивному програмуванні відбувається лише оновлення самих елементів сторінки, що пришвидшує роботу за рахунок зменшення трафіку.

Для впровадження принципів реактивного програмування в розробку веб-інтерфейсів можливе використання таких популярних інструментів:

- JQuery – старий але діючий фреймворк для оновлення елементів DOM-дерева без перезавантаження сторінки;
- React.js – прогресивна та потужна бібліотека для побудови інтерфейсів користувачів в манері реактивного програмування. Часто застосовується для випадків односторінкових додатків;
- Vue.js – сучасний фреймворк що базується на принципах React.js. Простий та зрозумілий.

Серед трьох варіантів був обраний останній. Vue.js є дуже комфортним коли розмова йде про створення невеликих, простих і швидких рішень. Інтерфейс дипломного проекту підпадає під такі критерії, бо не є основною метою даної дослідницької роботи.

Для створення головного сервісу, що забезпечує аналіз потокових даних була використана мова програмування Python. Це інтерпретована високорівнева мова програмування з динамічною типізацією, що може використовуватися у будь-яких сферах розробки. Свою популярність в основному мова отримала завдяки стислості коду, що забезпечується великою кількістю бібліотек, та використанню цієї мови в перспективній галузі машинного навчання та аналізу даних.

Для рішення поставленої задачі дипломного проекту, серед існуючих бібліотек мови Python знайшовся фреймворк, що дозволяє забезпечити швидку обробку великої кількості даних за допомогою розподілених обчислень. Його назва Apache Spark.

Apache Spark – це високошвидкісна технологія кластерних обчислень, що була розроблена на основі фреймворку Hadoop. На відміну від класичного обробника з ядра Hadoop, що реалізує дворівневу концепцію MapReduce [8] з дисковим сховищем, Spark використовує спеціалізовані примітиви для рекурентної обробки в оперативній пам'яті, завдяки чому дозволяє отримувати значний виграш в швидкості роботи для деяких класів задач, зокрема, можливість багаторазового доступу до завантажених в пам'ять даних робить бібліотеку привабливою для алгоритмів машинного навчання. Даний підхід в Spark називається кластерним обчисленням в пам’яті і є фундаментальним підходом цієї технології.

Іншою важливою складовою Apache Spark, що напряму зв’язана з обчисленнями в пам’яті, є структура даних Resilient Distributed Data (RDD) [9]. Коротко кажучи, RDD представляє собою незмінну розподілену колекцію об’єктів, що розроблена таким чином, щоб приховати значну частину

обчислювальної складності від користувачів. Вона об'єднує дані та розділяє їх по серверному кластеру, де вони можуть бути оброблені й далі або переміщені до іншого сховища даних, або пропущені через аналітичні моделі. Користувачеві не потрібно визначати, куди надсилаються певні файли або які обчислювальні ресурси використовуються для зберігання або отримання файлів.

RDD розшифровується як стійкий розподілений набір даних і відповідно до назви має такі характеристики:

- стійкість: оскільки Spark працює на кластері машин, втрата даних від апаратного збою є дуже актуальною проблемою, тому RDDs є відмовостійкими і можуть відновлювати себе в разі відмови;
- розподіленість: один RDD зберігається на ряді різних вузлів кластера, що не належать до жодного джерела (і жодної точки відмови). Таким чином, кластер може працювати на RDD паралельно;

Всі дані, з якими відбувається робота в Spark, зберігається в тій чи іншій формі RDD.

Кожне програмне забезпечення на основі Spark складається з одного модулю керувальника (Driver) і набору розподілених процесів виконавців (Executors).

Driver запускає метод main програми, планує виконання завдання за допомогою менеджера кластерів, аналізує, розраховує та розподіляє роботу між виконавцями.

Executor – це розподілений процес, що відповідає за виконання завдань. Кожен програмний засіб працюючий в рамках Spark має свій власний набір виконавців, які функціонують протягом життєвого циклу однієї програми Spark. Основні особливості виконавців:

- виконавці виконують всю обробку даних над роботами Spark;

- результати знаходяться в пам'яті, і зберігаються на диску тільки, якщо спеціально проінструктовані управляючою програмою;
- повертають результати драйверу після їх готовності;
- кожен вузол може мати від одного виконавця на один вузол до одного виконавця на ядро.

Звичайний процес програми Apache Spark має таку послідовність дій, що відбуваються за лаштунками:

1) програма стартує і ініціалізує її SparkContext. Лише після отримання SparkContext, програма може називатися Driver;
2) Driver вимагає від менеджера кластеру (Cluster Manager) ресурсів для запуску своїх виконавців;
3) менеджер кластерів запускає виконавців;
4) Driver запускає справжній код Spark;
5) виконавці запускають завдання і відправляють результати керувальнику;
6) SparkContext зупиняється і всі виконавці закриваються, повертаючи ресурси назад до кластера.

Даний процес зображений на рис. 2.1.

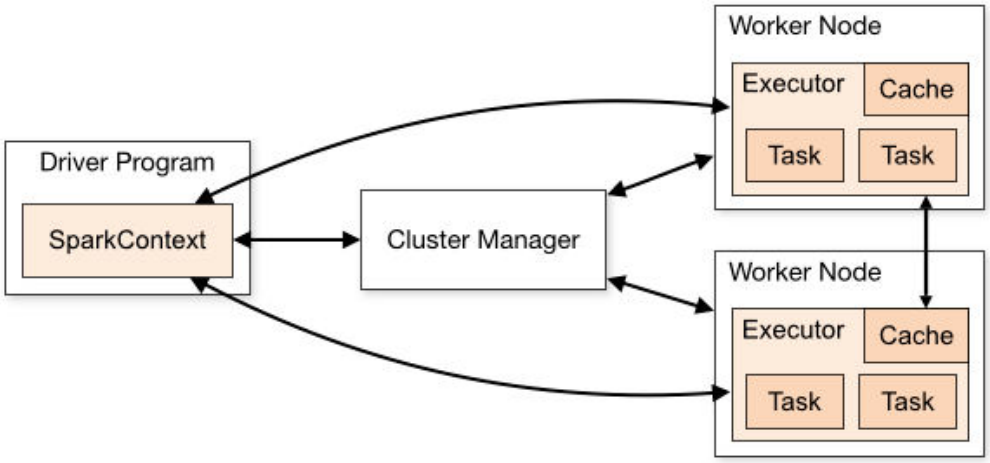


Рисунок 2.1 – Схема роботи програми Apache Spark

Весь функціонал, що забезпечує базові функції та механізми фреймворку структурно знаходиться в Apache Spark Core. Всі компоненти, що представлені в Spark базуються саме на цьому ядрі. Такими компонентами є:

- Spark SQL – модуль для роботи з структурованими даними;
- Spark Streaming – модуль для роботи з потоковими даними;
- MLib – бібліотека інструментів машинного навчання для Spark;
- GRaphX – модуль для роботи з графами та паралельною їх обробкою.

Візуальне зображення набору бібліотек Apache Spark наведено на рис. 2.2. Серед цих інструментів найцікавішим і найкориснішим для розробленого програмного засобу був Spark Streaming.

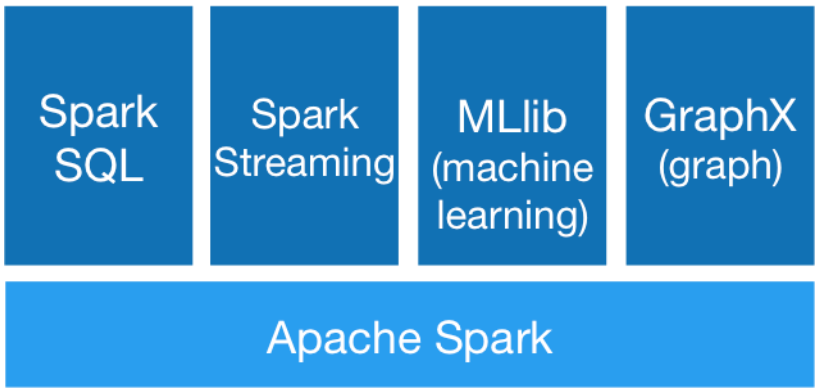


Рисунок 2.2 – Набір бібліотек Apache Spark

Spark Streaming є розширенням ядра Spark, що забезпечує масштабовану, високопродуктивну, стійку до відмов потокову обробку даних в реальному часі. Дані можуть надходити з багатьох джерел, таких як Kafka, Flume, файлів або TCP сокетів, і можуть бути оброблені за допомогою складних алгоритмів. Тобто можливе використання алгоритмів машинного навчання.

Під капотом Spark Streaming працює наступним чином. Модуль отримує потоки вхідних даних в реальному часі і розділяє їх на партії, які потім обробляються механізмами Spark для генерації остаточного потоку результатів також у вигляді партій. Даний процес проілюстровано на рис. 2.3.

Рисунок 2.3 – Послідовність обробки потокових даних в Spark Streaming

Spark Streaming користується абстракцією високого рівня, що називається дискретизованим потоком (discretized stream) або DStream, який представляє собою безперервний потік даних [10]. Внутрішньо DStream представлено у вигляді послідовності RDD.

Виявлення об’єктів як було зазначено в першому розділі повинно відбуватися з використанням існуючих методів глибинного навчання та глибинних нейронних мереж, а конкретно методики YOLO. Для забезпечення розподіленого аналізу даних за допомогою глибинних нейронних мереж в Apache Spark, в проекті використовується бібліотека BigDL.

BigDL – це бібліотека розподіленого глибинного навчання для Apache Spark. З її допомогою користувачі можуть писати свої програми з глибинним навчанням як стандартні програми Spark, що можуть безпосередньо працювати на кластерах Spark або Hadoop [11].

2.3. Представлення даних

Основним завдання розробленого програмного забезпечення є аналіз потокових відео даних. Потік отримується на стороні клієнта в бінарному форматі та надсилається покадрово до сервера через відкрите TCP з'єднання.

Далі на стороні сервера, дані з TCP сокета надходять до модуля Spark Streaming, де формуються в DStream, елементами якого є RDDs, що складаються з набору пересланих кадрів. Потім відбувається розподілена трансформація кожного кадру в RDD, а саме зміна їх розмірів до розмірів необхідних на вхід алгоритму виявлення та відповідно подальша передача даних на натреновану модель виявлення об'єктів YOLO.

Вихідними даними моделі є набір координат обмежувальної рамки об'єкту, впевненість в його класифікації та підпис назви класу для кожного з знайдених об'єктів на кадрі. Але так як аналіз відбувається розподілено в рамках контексту Apache Spark, то результатом обробки RDD в моделі виявлення об'єктів також є RDD, що складається з індивідуальних результатів аналізу моделі по кожному запису цієї RDD, тобто по кожному кадру. Далі отримане RDD знову проходить обробку для корекції результатів аналізу до оригінальних масштабів кадру.

Останнім етапом обробки даних в контексті Apache Spark є операція reduce, що в даному випадку збирає результати окремих записів RDD в результуючий. Він має вигляд звичайного списку результатів по кожному кадру в порядку їх первинного надходження до серверу. З цього списку формується JSON-формат відповіді, наведений на рис. 2.4, для пересилання через TCP з'днання до клієнта.

По отриманих результатах в веб-браузері формуються відповідні кадри з візуальним відображенням виявлених об'єктів, та виводяться в послідовності їх відправлення до сервера.

```
[
  {
    "label": ["person", "cell phone"],
    "conf": [0.9993208050727844, 0.994094967842102],
    "bbox": [
      [148, 54, 604, 456],
      [499, 117, 593, 311]
    ]
  },
  {
    "label": ["person", "cell phone"],
    "conf": [0.9989722967147827, 0.9955718517303467],
    "bbox": [
      [160, 56, 578, 452],
      [500, 115, 594, 315]
    ]
  },
  {
    "label": ["person", "cell phone"],
    "conf": [0.9707370400428772, 0.7312144637107849],
    "bbox": [
      [2, -6, 340, 480],
      [124, 280, 328, 480]
    ]
  }
]
```


Рисунок 2.4 – Результуючі дані аналізу RDD з трьох кадрів у JSON-форматі

де

label, conf, bbox – масиви результатів виявлення, елементи яких відповідно відносяться до даних про один об’єкт,

label – масив назв класів виявлених об’єктів,

conf – масив значень впевненості в правильності ідентифікації,

bbox – масив координат обмежувальних рамок: перші два значення відображають координати x та y верхнього лівого кутка рамки, інші два – x та y нижнього правого кутка.

Таким чином в розробленому програмному засобі виконується повний цикл обробки даних: від моменту їх отримання від сенсора до моменту їх візуалізації за результатами аналізу. Цикл може бути скорочений і стати неповним у разі використання розробки окремо від веб-інтерфейсу. В даному випадку сервіс виявлення об’єктів буде пропонувати свій прикладний

програмний інтерфейс (API), що на вхід отримуватиме потік кадрів, а на виході видаватиме результуючі дані в зазначеному вище форматі.

3. АЛГОРИТМ АНАЛІЗУ ПОТОКОВИХ ДАНИХ

В процесі проходження конвеєру в межах Spark Core, для виявлення об’єктів на кадрах відеопотоку був використаний алгоритм YOLO. Загалом обробка одного кадру відеопотоку відбувається в три основні етапи:

1) попередня обробка кадру;
2) аналіз алгоритмом виявлення об’єктів;
3) формування результатів після аналізу з врахуванням оригінального розміру кадру.

За попередню обробку кадру в даній роботі вважається процес зміни його розмірів до тих, які задовольняють алгоритм виявлення. Оригінальні кадри, що надходять з камер користувачів насправді можуть бути непередбачуваних розмірів, а алгоритм виявлення натренований на роботу лише з певними фіксованими. Конкретно в цій розробці алгоритм YOLO працює з розмірами зображень 448×448.

Для даної задачі були використані вже вбудовані методи бібліотеки BigDL. Написання власних алгоритмів в даному випадку є недоцільною, тим паче, що вже заготовлені методи бібліотеки є повністю оптимізованими і дають значно кращі показники ефективності ніж ті, що були б отримані написанням власних рішень.

Після проходження процесу попередньої обробки, нормалізований кадр далі передається до алгоритму виявлення об’єктів YOLO. В алгоритмі кадр в першу чергу потрапляє на CNN, приклад якої можна побачити на рис. 3.1, і на виході цієї CNN віддається тензор розміру $S * S * (B * 5 + C)$, якого достатньо для побудови рамок виявлених об'єктів. Тут S – розмірність сітки, B – кількість рамок, використовуваних в оцінці для кожної комірки, C – кількість класів, які нейронна мережа здатна розпізнавати.

Рисунок 3.1 – Схема використаної DNN

Алгоритм YOLO включає наступні етапи:

1) На зображення накладається сітка розмірністю $S * S$. Кожній комірці сітки відповідає вектор розмірністю $5 * B + C$, де число 5 визначає кількість показників кожної рамки. Використовувані показники: x, y – координати центру рамки всередині комірки (приймають значення від 0 до 1 по відношенню до розміру комірки); w, h – ширина і висота знайденої рамки (мають значення від 0 до 1 по відношенню до ширини і висоті вихідного зображення відповідно); c – ймовірність того, що рамка чітко визначена. Перші $5 * B$ значень вектора, відповідної комірки, характеризують саме ці параметри. Решта C значень, в цьому векторі показують ймовірності того, що центр об'єкта знаходиться в цій комірці. На даний момент ці ймовірності не прив'язані до рамок. Для знаходження ймовірностей класів для кожної з рамок необхідно помножити характеристику рамки c на ці C значень вектора, як показано на рис. 3.2.

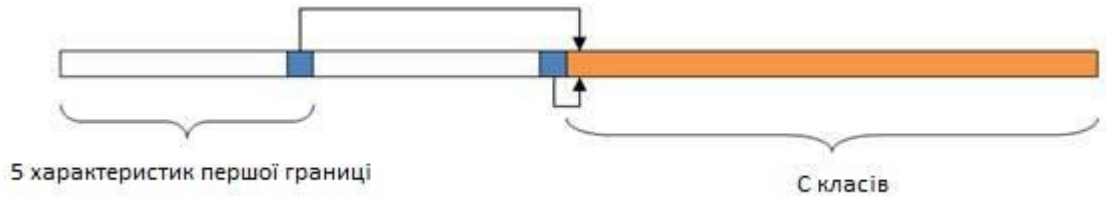


Рисунок 3.2 – Структура вектора відповідної комірки сітки

2) В результаті отримується $S * S * B$ рамок з ймовірністю класів. Для того щоб отримати підсумкове розпізнавання необхідно виконати наступне:
   a) фіксується який-небудь клас, для якого є вектор зі значенням ймовірності його класу з кожної з $S * S * B$ меж;
   b) обнуляється значення для тих рамок, у яких значення менше порогового (задається заздалегідь);
   c) сортується вектор по спаданню;
   d) застосовується алгоритм non maximal suppression (NMS). Працює він за таким принципом: на вході подається вектор з $S * S * B$ ймовірностей по певному класу для всіх рамок. Вибирається максимальне значення, так як вектор відсортований, то цей елемент знаходиться на першій позиції, і потім ця межа порівнюється з межами, розташованими правіше, у яких ймовірність по класу більше нуля. Порівняння відбувається за площею перетину: якщо площа перетину більше 0.5, то для кордону з меншою ймовірністю ця ймовірність обнуляється. За цим принципом порівнюються інші кордони. А потім фіксується наступна рамка з ненульовою ймовірністю і проводяться аналогічні операції порівняння. Таким чином, розглядаються всі рамки певного класу;
   e) далі береться наступний клас, і проводяться аналогічні операції порівняння і обнулення. Після подібної процедури отримуються

виряджені вектора з ймовірностями по кожному класу для кожної знайденої рамки.

Залишається тільки вирішити які рамки необхідно нанести на вихідне зображення. Метод відбору досить простий: розглядається кожна рамка, береться максимальне значення ймовірності по класах і, якщо воно більше нуля, то результати рамки вносяться в остаточний результат аналізу, інакше рамка пропускається і розглядається наступна.

У класичному випадку застосування алгоритму YOLO будується сітка розмірами 7 × 7, для кожної комірки будується по 2 рамки, і мережа навчається на 20 класах. На вхід нейронної мережі подається трьохканальне зображення розміром 448 × 448. Цей тензор пропускається через модифіковану мережу GoogLeNet (20 перших шарів) [15], на виході якої набір карт ознак просторовою розмірністю 14 × 14 × 1024. Далі застосовується набір згорток з ReLU (rectified linear unit) [22], який є функцією активації виду:

$$\varphi(x) = \begin{cases} x, \text{ для } x \geq 0, \\ 0.1 * x, \text{для } x < 0. \end{cases}$$

В результаті отримується тензор розміром 7 × 7 × 1024. На наступному етапі цей набір пропускається через повнозв'язний шар ReLU розмірністю 4096 × 1, а потім через повнозв'язний шар на виході якого отримується вектор 1470 елементів, який, перетворюється в тензор 7 × 7 × 30. Далі для цього тензора використовуються модифікації, описані вище.

В цьому проекті застосовується модифікована версія алгоритму YOLO - YOLOv2. Якщо перша версія алгоритму визначає координати обмежувальних рамок безпосередньо, використовуючи повнозв'язні шари поверх згорткового екстрактора ознак, то в YOLOv2 цей процес відбувається інакше, з використанням опорних прямокутників (anchor boxes) – стандартного набору прямокутників з певними значеннями довжини і ширини, які знаходяться на

етапі навчання за допомогою методу кластеризації k-means. Кожній комірці сітки зіставляються по 5 таких опорних прямокутників різного розміру, приклади яких продемонстровано на рис. 3.3, і замість прямого передбачення обмежувальної рамки мережа визначає відхилення від самого підходящого об'єкта опорного прямокутника. Такі відхилення обмежені, що дозволяє кожному опорному прямокутнику фокусуватися на об'єкті певної форми. Теж відбувається в разі, якщо центри різних об'єктів знаходяться в одній комірці: кожному з них ставиться відповідний опорний прямокутник.

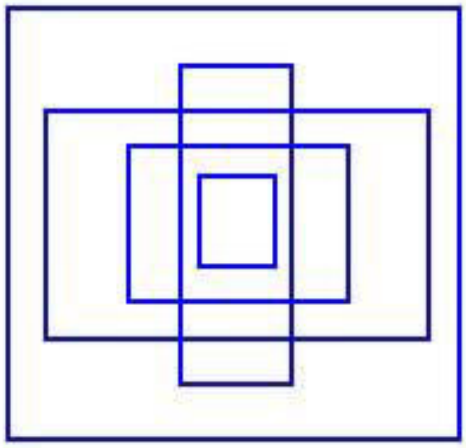


Рисунок 3.3 – Приклад опорних прямокутників для однієї комірки сітки

Дане нововведення дозволяє YOLOv2 позбутися повнозв'язних шарів, що раніше відповідали за визначення обмежувальних рамок, виводячи замість них результат, отриманий від згорткових шарів, тобто YOLOv2 перейшла на повністю згорткову нейронну мережу.

Більшість алгоритмів виявлення покладаються на згорткову нейронну мережу VGG-16 як на базовий екстрактор ознак [14]. VGG-16 – це потужна, точна класифікаційна мережа, але вона занадто надлишкова. Згорткові шари VGG-16 вимагають 30,69 мільярдів операцій з плаваючою точкою для одного проходу над одним зображенням з роздільною здатністю 224 × 224. Для підходу YOLOv2 використовується власна мережа на основі архітектури GoogLeNet

[15]. Ця мережа швидше, ніж VGG-16, використовуючи лише 8,52 мільярди операцій для прямого проходу. Однак її точність трохи гірше, ніж VGG-16.

Алгоритмом запропонована нова модель класифікації [16], яка використовується в якості бази YOLOv2. Як і в моделях VGG, тут використовуються в основному фільтри 3 × 3 і подвоюється кількість каналів після кожного етапу об'єднання [14]. Застосований глобальний середній пул для прогнозування, а також фільтри 1 × 1 для стиснення представлення ознак між згортками 3 × 3 [9]. Для стабілізації тренувань, прискорення збіжності і регуляризації моделі [7] використовується нормалізація партій. Остаточна модель, має назву Darknet-19, та має 19 згорткових шарів і 5 шарів maxpooling. Повна структура даної моделі зображена на рис. 3.4.

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Рисунок 3.4 – Структура згорткової нейронної мережі Darknet-19

Після завершення аналізу кадру алгоритмом YOLO, на виході отримується список результатів по кожному виявленому об’єкту. Для кожного об’єкта в результаті вказані наступні параметри:

- впевненість в правильності ідентифікації;
- назва класу виявленого об’єкта;
- координати центру рамки (приймають значення від 0 до 1 по відношенню до розміру відповідної комірки);
- параметри ширини та висоти (мають значення від 0 до 1 по відношенню до ширини і висоті вихідного зображення відповідно).

З цих даних на етапі кінцевої обробки формується результат, що може бути надісланий клієнту для відображення.

Основним завданням цього етапу є правильне перетворення координат центру рамки, ширини та висоти в координати верхнього лівого кутка та нижнього правого кутка рамок в оригінальному зображені. Маючи відносні параметри в площині модифікованого кадру, що подавався на вхід алгоритму, та оригінальні розміри кадру, легко отримати реальні значення цих параметрів. Далі знаючи числові значення координат центру рамки, ширини та висоти можна легко сформувати координати шуканих кутів рамок, використовуючи формули:

$$x_{tl} = x_0 - \frac{w}{2} \qquad y_{tl} = y_0 - \frac{h}{2}$$

$$x_{br} = x_0 + \frac{w}{2} \qquad y_{br} = y_0 + \frac{h}{2}$$

де

x, y – шукані координати кутів, де індекс tl – top left, br – bottom right,

x_0, y_0 – координати центру рамки,

w – ширина рамки,

h - висота рамки.

За повністю обробленими результатами по кадру формується кінцевий список з трьох елементів-масивів:

- масив назв класів для кожного знайденого об’єкта (рамки);
- масив упевненостей в правильності ідентифікації;
- масив знайдених координат кутів рамки для кожного виявленого об’єкта.

Параметри, що знаходяться під однаковими індексами в масивах належать до одного виявленого об’єкта.

На цьому алгоритм аналізу індивідуального кадру закінчується. Але даний процес в рамках Apache Spark виконується поетапно розподілено для всіх кадрів, що входять до поточного RDD. Такий підхід дозволяє значно пришвидшити обробку потокових даних.

4. ВИКОРИСТАННЯ РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

4.1. Використання через веб-інтерфейс

Використання розроблених програмних засобів відбувається через веб-інтерфейс. Користувачеві на сторінці доступні наступні два загальних типи елементів:

- елементи відображення результатів роботи системи;
- елементи керування ходом роботи системи.

До першого типу відносяться елемент-рамка з візуальним відображенням відеопотоку з камери користувача, та поле текстового повідомлення про поточний статус. При активованій роботі системи виявлення, тобто при постійному трафіку даних між клієнтом і сервісом, на рамці візуального відображення позначаються результати аналізу у вигляді трьох складових:

- нарисованої по отриманих координатах аналізу, обмежувальна рамка навколо об’єкта, що був виявлений;
- підпису над обмежувальною рамкою, що складається з відсоткового значення впевненості моделі в правильності виявлення, та назви класу об’єкта, що був виявлений;
- числового значення в кутку відео рамки, що визначає кількість оброблених кадрів на секунду (FPS).

Поле текстового повідомлення слугує для інформування користувача. В різні етапи роботи програми може мати різні повідомлення:

- «Click start button to begin object detection» - у випадку коли робота системи, ще не була запущена;
- «Objects detected» - у випадку працюючої системи та виявлення об’єкта;

- «No objects detected» - у випадку працюючої системи та не виявлення об’єкта;
- «Click start button to begin detection again» - у випадку не працюючої системи після натискання кнопки «Stop».

Друга категорія елементів інтерфейсу представляє собою засоби управління ходом роботи системи виявлення. Категорія представлена двома керуючим кнопками та випадаючим списком доступних для виявлення об’єктів.

Дві кнопки «Start» і «Stop» необхідні відповідно для запуску та зупинки аналізу відеопотоку сервісом виявлення. У списку доступних для виявлення об’єктів можна обрати для якого конкретно типу об’єкта буде відбуватися виявлення. За замовчуванням виявлення буде відбуватися для всіх класів об’єктів списку.

Наповненість списку елементами залежить від об’єктів, які може виявляти натренована модель розробленої системи. Перелік назв класів цих об’єктів зберігається в конфігураційному файлі, та надсилається на клієнт разом з відкриттям веб-сторінки.

Для демонстрації роботи проекту модель була натренована за допомогою набору даних Common Objects in Context (COCO) [12]. COCO – це великий набір даних для завдань сегментації, виявлення, класифікації, що включає в себе:

- 330 тис. зображень (більше 200 тис. з яких підписані)
- 1.5 мільйон об’єктів
- 80 категорій об’єктів
- по 5 об’єктів на 1 зображенні

Для забезпечення виявлення нових об’єктів за допомогою цього програмного засобу, необхідно перетренувати модель YOLO на їх виявлення, та внести назви категорій до конфігураційного файлу. Після цього користувач зможе запускати систему для виявлення цих об’єктів та користуватися можливостями випадаючого списку.

4.2. Аналіз результатів

Для аналізу результатів роботи системи необхідно встановити параметри, які будуть прийняті як показники ефективності. У двійковій класифікації використовується показник середньої точності (AP) як головний показник, що описує криву точності-повноти класифікаторів. Але конкретно для задач виявлення об'єктів найчастіше використовується інша метрика – середня AP (mAP). Це просто середнє значення AP, обчислених по всіх класах тесту.

Іншою важливою метрикою є кількість оброблених кадрів на секунду (FPS). Чим більше цей показник тим реальніше застосування системи для потокового аналізу даних в справжніх умовах.

Для визначення ефективності роботи програмного засобу даного дипломного проекту не була використана метрика mAP. Аргументацією такого рішення стало те, що в тестуванні брали участь системи, що основані на одній і тій самій моделі виявлення об’єктів, і показник mAP не змінюється від засобів розподілення навантаження. Основною характеристикою, що цікавила при дослідженні роботи розробленої системи стала та, що показує швидкість обробки потоку даних, тобто метрика FPS. Її значення і буде показником ефективності системи.

Оцінка була проведена між декількома наступними варіаціями системи:

- класичної системи, що не має застосування розподілених обчислень.
- розробленої системи з використанням n-ядер CPU як вузлів кластеру, де n – має значення 2, 4, 6 та 8;

Визначення FPS в тестуванні відбувалося наступним чином:

1) на початку тестування ініціалізується лічильник проаналізованих кадрів із значенням 0;
2) протягом 60 секунд надається потік кадрів для аналізу та збільшується лічильник після завершення аналізу кожного кадру;

3) вираховується результуюче значення FPS як середнє по формулі:

$$FPS = \frac{counter}{time}$$

де

FPS – результуюче середнє значення кількості кадрів на секунду,

$counter$ – значення лічильника оброблених кадрів,

$time$ – час проведення тестування в секундах, що в даному випадку має значення 60.

Для надання об’єктивних результатів роботи всі варіації, що тестувалися, були поставлені в однакові умови, не враховуючи кількість вузлів кластера:

- для виявлення, в рамках симуляції аналізу в реальному часі, був наданий один і той же відеопотік у вигляді заздалегідь вибраного потоку кадрів;
- обчислення відбувались почергово на одній і тій же машині (8-ядерний процесор Intel Core та 24 ГБ оперативної пам’яті);
- за алгоритм виявлення об’єктів був взятий YOLOv2.

Тестування проводилося в локальному режимі, тобто як вузли серверного кластеру використовувалися ядра процесора. При чому для модулю Spark Streaming фреймворку Apache Spark окремо необхідне використання одного ядра процесору для роботи, так званого, отримувача (Receiver), для оброблення вхідного потоку даних. Таким чином фактична кількість вузлів кластера менша від заявленої на одиницю.

Результати роботи систем відповідно наведені в табл. 4.1.

Система, що тестувалася	Кадри на секунду (FPS)
Система без застосування розподілених обчислень	24
Система з використанням 2-х локальних вузлів кластера	22
Система з використанням 4-х локальних вузлів кластера	29
Система з використанням 6-х локальних вузлів кластера	34
Система з використанням 8-ми локальних вузлів кластера	45

Таблиця 4.1 – Результати тестування системи

Як одразу видно з результатів проведеного тестування, система надала позитивні показники. З додаванням розподілених обчислень до існуючого алгоритму YOLOv2, кількість кадрів, що були оброблені за секунду збільшились в порівнянні з системою без модифікацій. Крім цього, збільшення кількості вузлів кластеру надала позитивне зростання швидкодії системи, що також видно з побудованого графіку граф. 4.1.

Виокремити тільки можна випадок коли під роботу розробленого програмного засобу надано всього 2 ядра. Видно, що результати трохи гірші ніж у системи без модифікацій. Пояснюється така поведінка специфікою локального тестування, а саме специфікою технології Spark Streaming, що застосовує окремо одне ядро для своїх потреб. Таким чином при двох ядрах, фактично працює лише одне. Даної проблеми не буде при використанні реальних вузлів серверного кластера.

Графік 4.1 – Тренд росту швидкодії від кількості вузлів кластеру в локальному режимі

Аналізуючи конкретні значення показників тестування, можна зазначити, що на апаратному забезпеченні де відбувалося тестування, вони виявилися не високими. Це пов’язано з декількома факторами:

1) виконання обчислень на основі глибинних нейронних мереж на CPU значно повільніше ніж на GPU;
2) складові системи де відбувалося тестування не були достатнього рівня для забезпечення високої швидкодії рішення подібних задач;
3) як вузли кластеру були використані лише локальні ядра CPU.

Не зважаючи на відносно невисокі показники, що випливають з перелічених причин, проведене тестування показало шукану тенденцію росту продуктивності. Переносячи даний програмний засіб на системи більших потужностей, пропорційно виростуть і показники ефективності. Такими системами можуть бути системи, що усувають недоліки даної системи де проводилося тестування.

В першу чергу, замість використання локальних розподілених обчислень повинен бути введений повноцінний серверний кластер з окремими машинами в ролі його вузлів. Кожна машина кластеру повинна мати передове обладнання, що включає найновіші процесори та оперативну пам'ять, що за об’ємом не менше 24 ГБ. Крім цього можна замість обчислень на CPU ввести обчислення на основі GPU, що є нормальною практикою в сфері аналізу Big Data.

ВИСНОВОК

Програмний продукт дипломного проекту за результатами тестування задовольнив поставлені до нього вимоги. Аналіз даних в потоковому режимі з виконанням важких обчислень, зокрема виявленням об’єктів на основі глибинного аналізу є витратним процесом. Але знайдене рішення з застосуванням розподілених обчислень дозволило зменшити навантаження та пришвидшити аналіз даних в реальному часі.

Важливим опорним пунктом розробленого програмного засобу стало використання існуючого підходу виявлення об’єктів YOLOv2, який при попередньому аналізі аналогів показав найкращі показники для систем працюючих в реальному часі. Крім цього підходу також можлива модифікація даної розробки за допомого інших алгоритмів, зокрема YOLOv3 та Single Shot Detector (SSD), які мають свої певні переваги над використовуваним алгоритмом. SSD може збільшити роздільну здатність виявлення, а YOLOv3 точність ідентифікації, але за рахунок можливого зменшення швидкодії.

Крім переваг розробленої системи є й її недоліки. До недоліків відносяться:

- залежність точності системи від конкретного алгоритму виявлення;
- проведення обчислень за допомогою CPU, а не кластеру GPUs;
- необхідність використання серверного кластеру, що може складатися з великої кількості машин.

Не зважаючи на недоліки, система має достатній рівень, щоб бути застосованою в реальних комерційних умовах.

Крім цього, розподілення обчислень таким чином дозволяє обробляти значно більші об’єми даних за умови розширення серверного кластеру. Зокрема, даний продукт може бути використаний для одночасного оброблення потоків даних з декількох відеокамер, при відповідному внесенні невеликих

модифікацій в код системи. Це значно розширить галузь застосування розробленого програмного засобу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

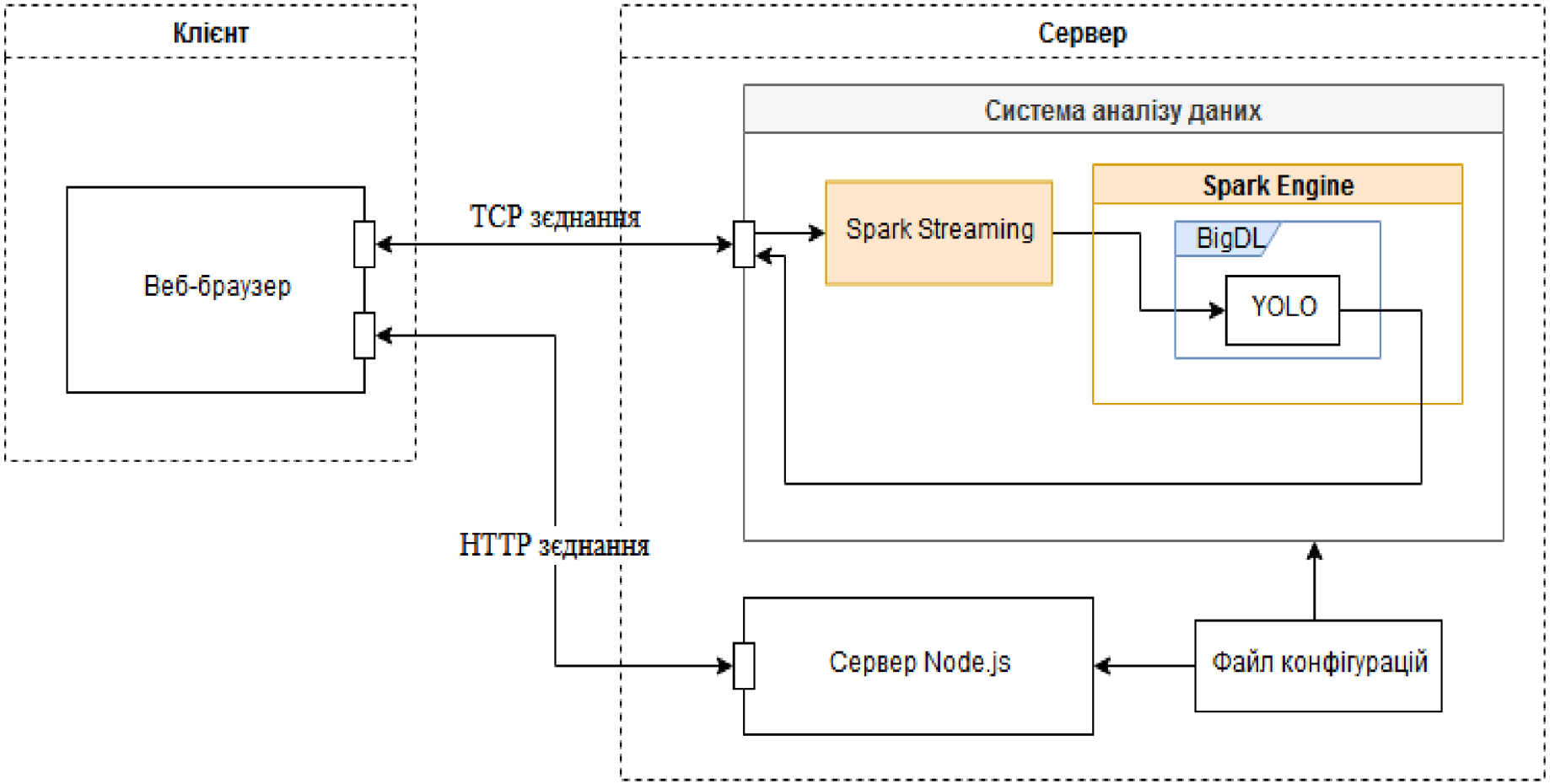
1. P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, p. 1627, 2010.
2. R. Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, arXiv: 1311.2524, 2013.
3. R. Girshick, “Fast r-cnn,” in ICCV, 2015.
4. S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015,
5. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection”, in CVPR, 2016.
6. J. Redmon, A. Farhadi, “Yolo9000: better, faster, stronger”, arXiv: 1612.08242, 2016.
7. J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement”, arXiv, 2018.
8. Jeffrey Dean, et al. “MapReduce: simplified data processing on large clusters”, OSDI 2014.
9. Matei Zaharia, et al. “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing”, NSDI, 2012.
10. Matei Zaharia, et al. “Discretized Streams: Fault-Tolerant Streaming Computation at Scale”, SOSP 2013.
11. J. Dai, Y. Wang, “BigDL: A Distributed Deep LearningFramework for Big Data”, arXiv: 1804.05839, 2018.
12. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in ECCV, 2014.

13. Jason (Jinquan) Dai, et al. “Building Large-Scale Image Feature Extraction with BigDL at JD.com”, https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom.
14. K. Simonyan, A. Zisserman. “Very deep convolutional networks for large-scale image recognition”, arXiv: 1409.1556, 2014.
15. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich., “Going deeper with convolutions”, CoRR, abs/1409.4842, 2014.
16. J. Redmon, “Darknet: Open source neural networks in c”, http://pjreddie.com/darknet/, 2013–2016.
17. C. Cortes and V. Vapnik, “Support vector machine,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
18. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models”, IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, pp. 1627–1645, 2010.
19. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
20. M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge 2007 (voc 2007) results (2007)” 2008.
21. J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” Int. J. of Comput. Vision, vol. 104, no. 2, pp. 154–171, 2013.
22. W. Shang, K. Sohn, D. Almeida, and H. Lee, “Understanding and improving convolutional neural networks via concatenated rectified linear units,” in ICML, 2016.
23. N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in CVPR, 2005.

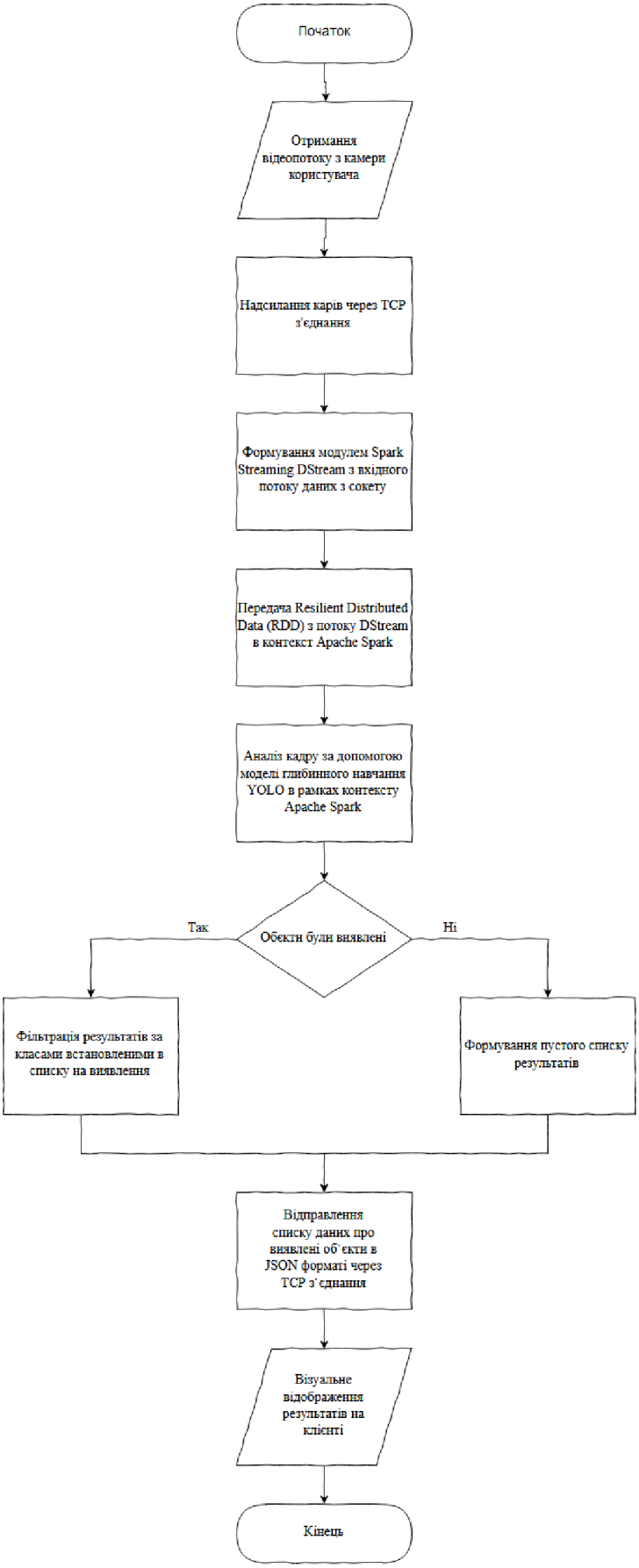
24. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in ECCV, 2016.

25. R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection” in ICIP, 2002.

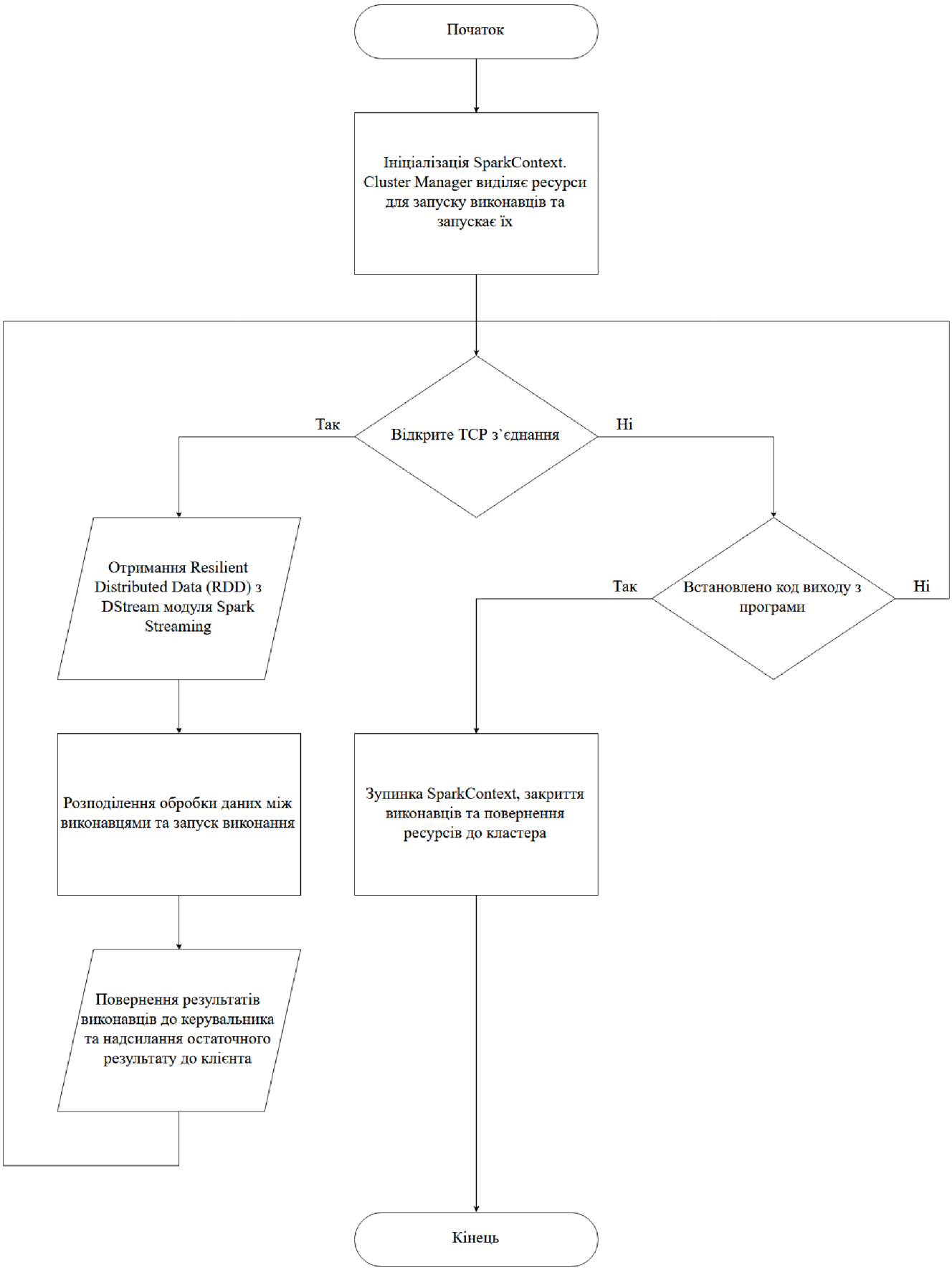
Додаток 1
Копії графічних матеріалів



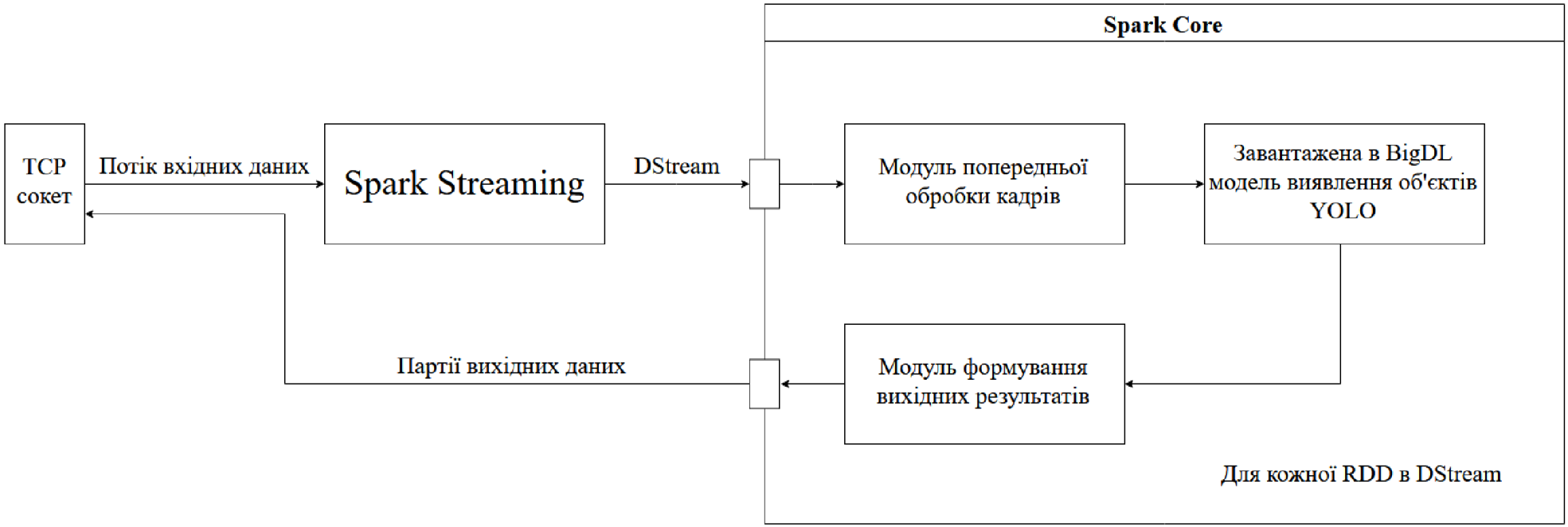
					ІАЛЦ.045440.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних	Літ / Маса / Масштаб
Розроб		Тимошенко І.О.				
Перев		Петрашенко А.В.				
Т.контр					Архітектура програмного засобу Схема структурна	Аркуш 1 / Аркушів 1
Н.контр		Клятченко Я.М.				
Затв		Тарасенко В.П.				КПІ ім. Ігоря Сікорського, ФІМ, КВ-51



					ІАЛЦ.045440.006 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб		Тимошенко І.О.			Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних	Літ / Маса / Масштаб
Перев		Петрашенко А.В.				
Т.контр					**Робочий потік даних. Схема алгоритму**	Аркуш 1 / Аркушів 1
Н.контр		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФІМ, КВ-51
Затв		Тарасенко В.П.				



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.007 ДЗ		
Розроб		Тимошенко І.О.			Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних	Літ	Маса	Масштаб
Перев		Петрашенко А.В.						
Т.контр					**Аналіз даних в контексті Apache Spark** **Схема алгоритму**	Аркуш 1	Аркушів 1
Н.контр		Клятченко Я.М.				КПІ ім. Ігоря Сікорського, ФІМ, КВ-51	
Затв		Тарасенко В.П.					



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб		Тимошенко І.О.		
Перев		Петрашенко А.В.		
Т.контр				
Н.контр		Клятченко Я.М.		
Затв		Тарасенко В.П.		

ІАЛЦ.045440.008 Д4

Веб-орієнтовані програмні засоби аналізу потокових сенсорних даних

Структура сервісу обробки даних
Схема структурна

Літ	Маса	Масштаб

Аркуш 1	Аркушів 1

КПІ ім. Ігоря Сікорського, ФІПМ, КВ-51